StrucOpt manuscript No.
(will be inserted by the editor)

Multilevel parallel optimization using massively parallel structural dynamics*

M.S. Eldred, A.A. Giunta, and B.G. van Bloemen Waanders

Abstract A large-scale structural optimization of an electronics package has been completed using a massively parallel structural dynamics code. The optimization goals were to maximize safety margins for stress and acceleration resulting from transient impulse loads, while remaining within strict mass limits. The optimization process utilized nongradient, gradient, and approximate optimization methods in succession to modify shell thickness and foam density values within the electronics package. This combination of optimization methods was successful in improving the performance from an infeasible design which violated response allowables by a factor of two to a completely feasible design with positive design margins, while remaining within the mass limits. In addition, a tradeoff curve of mass versus safety margin was developed to facilitate the design decision process. These studies employed the ASCI Red supercomputer and utilized multiple levels of parallelism on up to 2560 processors. In total, a series of calculations were performed on ASCI Red in five days, where an equivalent calculation on a single desktop computer would have taken greater than 12 years to complete.

Key words parallel optimization, multilevel parallel computing, structural dynamics

Received: October XX, 2002

M.S. Eldred, A.A. Giunta, and B.G. van Bloemen Waanders

Optimization and Uncertainty Estimation Department
Sandia National Laboratories
Albuquerque, NM 87185–0847, USA
e-mail: mseldre@sandia.gov, aagiunt@sandia.gov, bartv@sandia.gov

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

* Presented as paper 2001–1625 at the 42nd AIAA-/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Seattle, WA, April 16-19, 2001

1 Introduction

This report describes the design optimization of an electronics package (EP) which is one component of an atmospheric re-entry vehicle. The design study was performed in the Spring of 2000 using massively parallel, high-fidelity structural dynamics simulations conducted on the Accelerated Strategic Computing Initiative Option Red supercomputer (ASCI Red) at Sandia National Laboratories.

During a five-day period, a block of up to 2560 processors on ASCI Red was employed to run up to 10 concurrent structural dynamics simulations, each employing 256 processors. Approximately 500 of these structural dynamics simulations were performed, each of which involved a transient structural analysis for a 500,000 degree-of-freedom (DOF) finite element model. The set of calculations performed during this five-day period would have required more than *twelve years* of computation time on a single desktop computer. While this use of up to 2560 processors for twelve processor years is on the extreme end of what is feasible with current computational resources, it is anticipated that such large-scale design optimization studies will become more commonplace as massively parallel computers and scalable parallel simulation codes become production computing tools in the aerospace, automotive, and biomedical fields. Indeed, recently published work performed for aerospace (Biros and

Ghattas (2000)) and automotive applications (Yang *et al.* (2000); Sobieszczanski-Sobieski *et al.* (2001)) demonstrates that parallel computing is becoming a standard tool in the vehicle design process. The maxim of Computerized Parkinson's Law (Thimbleby (1993)) states that simulation complexity tends to increase to fill the available resources, and Venkataraman and Haftka (2002) suggests that analysis complexity and computing power have historically increased in direct proportion due to practical requirements on analysis turnaround time. The authors submit that this increasing appetite for higher fidelity models naturally leads to massively parallel simulation and multilevel parallel computing, the subjects of this report.

Optimization problems of this complexity and computational expense pose many technical challenges. For good computational efficiency, the structural dynamics and optimization codes must be scalable to a large numbers of processors (order $10^2 - 10^4$). For the structural dynamics software, this entails the use of specific numerical techniques (e.g., iterative linear solvers, cache-optimized BLAS (Heath (1997))) that exploit both the structure of the finite element model and the hardware configuration of the parallel computer. For the optimization software, parallel scheduling of simulations must also exploit the hardware configuration of the parallel computer and should be robust with respect to heterogeneities in the scheduled jobs. In addition, the optimization software should be fault-tolerant with respect to simulation and hardware failures and should be robust to nonsmooth response variations generated from the simulations.

This paper provides the details of a production design effort and, as a result, has an application emphasis. The intent is to investigate optimization tools for a large-scale engineering application, to convey lessons learned, and to provide verification of the multilevel parallel computing techniques described in Eldred *et al.* (2000). Over the course of the study, nongradient-based, gradient-based, and approximate optimization methods were applied in an iterative, evolving process in order to improve the design of the EP. The results of this study are not intended to compare methods, but rather to demonstrate the utility of having a "toolbox" of sensitivity analysis and optimization algorithms from which one can tailor the optimization procedures as more is learned about the features of a particular application.

Sections 2-5 provide background information on the electronics package model, the Salinas structural dynamics software, the DAKOTA optimization toolkit, and the ASCI Red supercomputer, respectively. Section 6 describes the formulations, methods, and results in the EP design optimization problem, and Section 7 provides concluding remarks.

2 Electronics Package Model

The motivation for the optimization study was to help designers improve the structural integrity of a new EP structural design concept. Since this EP design was a refurbishment for the re-entry vehicle, it provided the opportunity to incorporate several new components into the existing package. However, an important requirement was to avoid changing the flight characteristics of the re-entry vehicle, so a restriction of no more than 10% deviation from the nominal EP mass was imposed. In order to add functionality but maintain mass, the EP design concept replaced some metallic support structure with rigid support foam. Thus, the design problem is a challenging one in that an EP design concept with less structural support must still survive high stresses and accelerations from severe re-entry vehicle structural loading conditions. A solid model of the EP design concept is shown in Figure 1.

Over time, the level of fidelity in structural dynamics analysis has increased significantly (Figure 2) as a result of more advanced computers and, most recently, the availability of a massively parallel structural dynamics code. This has allowed for the inclusion of more geometric detail in the computational models, which results in more geometrically accurate and predictive models, more numerically converged finite element results, and a reduction in the need for analysts to perform ad hoc model simplifications.

In this study, the EP geometry was discretized using a finite element model having 500,000 degrees of freedom (DOF). This model captures most of the design features of the EP and provides sufficient detail for the optimization study. Using 256 processors on ASCI Red, a single transient structural analysis of this model required approximately 40 minutes. While larger finite element models of the EP have been created (with more than 10 million DOF), these large models were considerably more expensive and were deemed to be impractical for use in this optimization study.

The 500,000 DOF finite element model was constructed using 55 geometric blocks where each block corresponds to one or more subcomponents inside the EP. Some of these blocks were structural shell

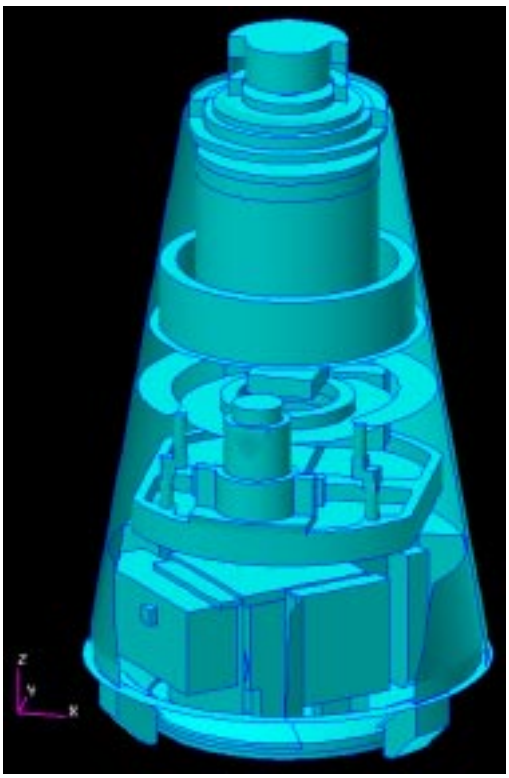

Fig. 1 A CAD model of the electronics package.

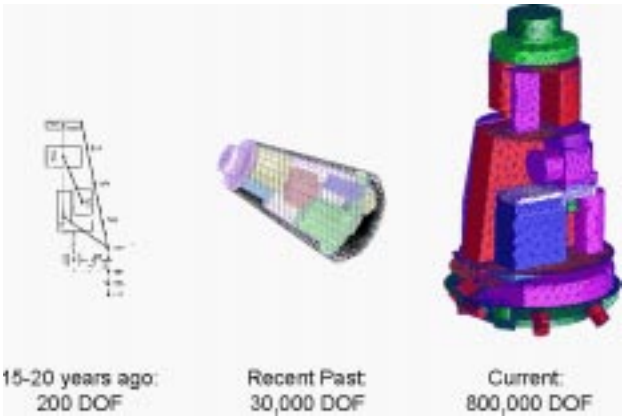

Fig. 2 Historical progression of the electronics package finite element model fidelity.

elements within the EP, while others were regions of foam encapsulant used to cushion the EP subcomponents.

The design variables for this study were the shell thicknesses of a subset of the structural blocks, and the density values for a subset of the foam encapsulant blocks. These design variables were selected based on a modal sensitivity analysis, with those blocks having the largest impact on the first 100 frequencies (greatest number of frequency derivatives exceeding a threshold) being selected as design parameters. While modal analysis is not a direct component of this design problem, this parameter screening approach is effective in identifying parameters which have global influence on model results, and the results were consistent with the engineering judgment of the EP analysts.

The computational simulation models the effect of a transient impulse loading event on the EP. Structural response was computed over a time duration of three milliseconds using 300 equal time steps. Response quantities of interest were the mass of the EP, along with the maximum stress and acceleration values within each of the 55 blocks (maximum over all DOF in a block for all time steps). These mass, stress, and acceleration quantities were used in the objective function and constraints in order to formulate the design problem.

For parallel processing, the EP finite element model underwent domain decomposition to separate it into 256 subdomains, i.e., one subdomain for each processor. These subdomains were selected by the domain decomposition software based on parallel load balancing considerations, and did not in general correspond to any geometric subcomponent boundaries in the EP.

3 Salinas: Massively Parallel Structural Dynamics

Salinas (Reese *et al.* (2000)) is a general-purpose, finite element structural dynamics code designed to be scalable on massively parallel computers. Currently, the code offers static analysis, direct implicit transient analysis, eigenvalue analysis for computing modal response, and modal superposition-based frequency response and transient response. In addition, semi-analytical derivatives of many response quantities with respect to user-selected design parameters are available. Salinas also includes an extensive library of standard one-, two-, and three-dimensional elements, nodal and element loading, and multipoint constraints. Salinas solves systems of equations using an iterative, multilevel solver, which is specifically designed to exploit massively parallel computers.

The linear solver used by Salinas was selected based on the criteria of robustness, accuracy, scalability and efficiency. Neither direct methods (e.g., sparse Gaussian elimination) nor general purpose iterative solvers (e.g., the preconditioned conjugate gradient method with over-lapping Schwartz pre-

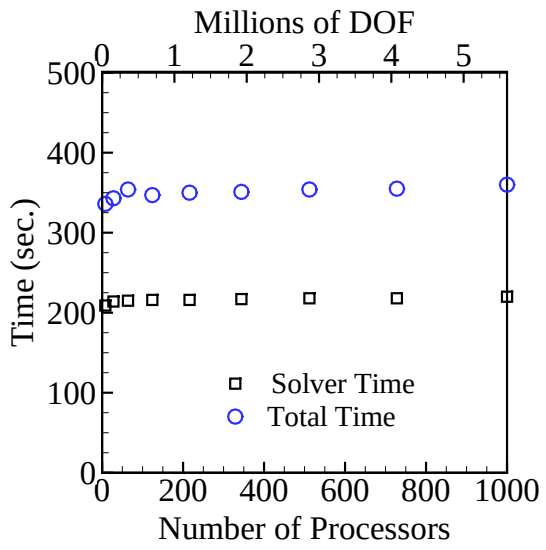

Fig. 3 Scalability for FETI (solver time) and Salinas (total time).

conditioner available in Aztec (Tuminaro *et al.* (1999))) perform well for parallel solution of linear systems obtained from the discretization of structures using high order plate and shell elements. In this case, the underlying partial differential equation is the fourth order biharmonic equation for which special purpose iterative solvers are necessary. This led to the selection of a multilevel domain decomposition method, Finite Element Tearing and Interconnect (FETI) (Farhat and Roux (1992)), that is the most successful parallel solver known to the Salinas developers for the linear systems applicable to structural mechanics. FETI is a mature solver, with some versions used in commercial finite element packages such as ANSYS (O'Neal and Murgie (2002)). As shown in Figure 3, FETI is scalable in the sense that, as the number of unknowns increases and the number of unknowns per processor remains constant, the time to solution does not increase. Further, FETI is accurate in the sense that the convergence rate does not deteriorate as the iterates converge.

An eigensolver was selected for Salinas based on these same criteria: robustness, accuracy, scalability and efficiency. Both a Lanczos-based solver (Day (1998)) and subspace iteration were evaluated. The Lanczos algorithm solves the minimal number of linear systems required to approximate a set of modes to a given accuracy, and Lanczos-based methods are significantly more efficient than subspace iteration. PARPACK (Maschhoff and Sorensen (1996)) is a scalable Lanczos-based solver that was selected because its memory usage is minimal, the software is reliable, and the number of linear systems solved per mode is nearly minimized.

4 DAKOTA: Multilevel Parallel Optimization

The DAKOTA (Design Analysis Kit for Optimization and Terascale Applications) toolkit (Eldred *et al.* (2002a,b,c)) is an open source software framework that provides a flexible and extensible interface between simulation codes and iterative systems analysis methods. DAKOTA contains algorithms for optimization with gradient and nongradient-based methods; uncertainty quantification with sampling, analytic reliability, and stochastic finite element methods; parameter estimation with nonlinear least squares methods; and sensitivity/primary effects analysis with design of experiments and parameter study capabilities. These capabilities may be used on their own or as components within advanced strategies such as surrogate-based optimization, mixed integer nonlinear programming, or optimization under uncertainty. DAKOTA provides generic simulation interfacing facilities which allow the use of a variety of engineering and physics simulation codes as "function evaluations" within an iterative loop. DAKOTA manages the complexities of its analysis and optimization capabilities through the use of object-oriented abstraction, class hierarchies, and polymorphism (Stroustrup (1991)). The flexibility of the framework allows for easy incorporation of the latest external and internal algorithmic developments.

Parallelism is an essential component of the DAKOTA framework. Particular emphasis has been given to simultaneously exploiting parallelism at a variety of levels in order of achieve near-linear scaling on massively parallel computers. For example, DAKOTA can manage concurrent optimizations, each with concurrent function evaluations, each with concurrent analyses that utilize multiple processors. Eldred *et al.* (2000) provides guidance on how to select partitioning schemes and scheduling algorithms within these levels in order to maximize overall parallel efficiency and to ensure robustness with respect to heterogeneity (e.g., variability in simulation duration). A common case is two levels of parallelism, in which concurrent function evaluations each run on multiple processors. In this study, DAKOTA employed two levels of parallelism by managing up to 10 concurrent Salinas invocations, each of which employed 256 compute nodes. Through this combination of coarse-grained and fine-grained parallel com-

puting, DAKOTA was able to effectively utilize up to 2560 processors and achieve rapid turnaround on this large-scale design study.

5 ASCI Red Supercomputer

For this optimization study, substantial computational resources were required. Within Sandia National Laboratories, one of the primary production computing platforms is the ASCI Red supercomputer (Mattson and Henry (1997); Tomkins (1996)).

5.1 Architecture

ASCI Red is a massively parallel, distributed memory, multiple input multiple data (MIMD) computer. It has a peak performance of greater than three TeraFLOPS (trillion floating point operations per second). It is designed so that file input/output (I/O), memory, disk capacity, and communication are scalable. Standard parallel programming libraries, such as the Message Passing Interface (MPI) (Snir *et al.* (1996)), make it relatively straightforward to port parallel applications to this system.

The processors in the ASCI Red supercomputer are organized into four partitions: compute, service, system, and I/O. Of these, the service partition provides support for interactive users, application development, and system administration. This partition runs a full UNIX operating system. The parallel applications execute in the compute partition, which contains nodes optimized for floating point performance and for high bandwidth communication. This partition executes the Cougar operating system (Greenberg *et al.* (1997)) which is a lightweight kernel intended to leave as much node memory as possible available for the application. Each compute node consists of two 333 MHz Intel Pentium-II Xeon Core processors with 256 MBytes of RAM. In this study, only one processor per node was used for computation while the other processor was used for communication, although a new "virtual node" capability allows the use of both node processors for computation. The system hardware and performance attributes of ASCI Red are summarized in Table 1.

Table 1 Hardware and performance characteristics of the ASCI Red supercomputer.

Compute Nodes	4510
Service Nodes	52
System and I/O Nodes	87
Total Processors	9298
System RAM (TeraBytes)	1.2
Compute Node Peak Performance (MegaFLOPS)	666
System Peak Performance (TeraFLOPS)	3.1
System Linpack Performance (TeraFLOPS)	2.4

5.2 Salinas/DAKOTA Implementation on ASCI Red

DAKOTA can be interfaced with simulation codes in a variety of ways depending on the level of intrusiveness one is willing to support, on the desired performance, and on the underlying compute architecture. The simplest approach is the "black-box" method, which employs process creation facilities such as C system calls (Kernighan and Ritchie (1988)) or UNIX forks (Glass (1993)). This is the least intrusive method in that the simulation can be used as is, with no modifications. It is also the least efficient method since it incurs the overhead of creating separate processes for the simulations. In practice, this overhead is usually small relative to the expense of the simulations. The most computationally efficient interface technique is the "direct" method in which the simulation code (e.g., Salinas) is linked into DAKOTA as a callable function. While efficient, the direct interface is intrusive to the simulation code since the code must be transformed to a subroutine and, in the parallel case, made modular on an MPI communicator. In addition, it complicates the use of pre- and post-processing tools (e.g., mesh generation, domain decomposition and reconstitution) since direct interfaces to these tools are required as well.

These two interfacing approaches have additional distinctions when applied on massively parallel computers which employ a service/compute node design. In particular, the black-box approach involves the execution of DAKOTA on the service nodes where it creates concurrent simulation driver processes on the service nodes. Each of these simulation drivers then launches a parallel simulation into the compute node partition. DAKOTA must then continuously monitor for the completion of these simulations, again utilizing service node resources. The direct approach, on the other hand, involves the ex-

ecution of a combined executable on the compute nodes only. The management of concurrent multiprocessor simulations is performed internally using MPI communicators. Consequently, the direct approach places fewer demands on the service partition than the black-box approach.

For this study, a black-box approach using system calls was selected, which allowed the use of a separate, unmodified Salinas executable. In this case, DAKOTA was run on the service node partition where it coordinated concurrent Salinas jobs on the compute partition. This is depicted in Figure 4. A key component of conducting a study of this type in a shared resource environment (i.e., in the presence of NQS/PBS job queues) was the ability to make a single request for a large block of processors (e.g., 2560 processors) and then schedule sets of smaller parallel jobs (e.g., 10 concurrent jobs of 256 processors each) within partitions of the larger allocation. This avoided the repeated queue delays that would otherwise have occurred if the smaller jobs were queued separately. In addition, pre- and post-processing steps were important components for allowing communication between DAKOTA and Salinas. Values of the design variables were written by DAKOTA to a file and then incorporated into the Salinas input file using a Sandia-developed file parsing program. The output of Salinas was post-processed to provide the mass and safety margin data values needed for the optimization studies. While the results from each domain-decomposed, parallel simulation could be gathered into a single file for post-processing, it was more expedient to evaluate safety margins across separate subdomain databases. This entire cycle was automated using a single UNIX driver script that was invoked by DAKOTA. While DAKOTA was executed on a single service node and each of the system calls to concurrent Salinas drivers were initiated from this single service node, a resident load spreading utility relocated Salinas monitoring processes among the entire service partition in order to distribute the application load.

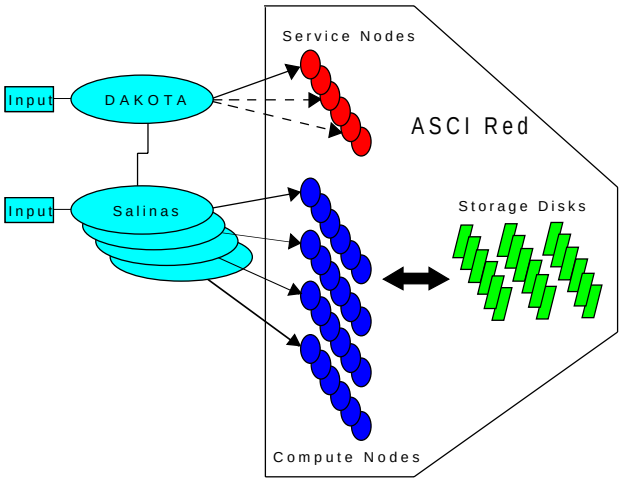

Fig. 4 A depiction of the DAKOTA/Salinas implementation on ASCI Red.

5.3 Computational Issues

Optimization studies which create multiple simulation processes impose different loads on supercomputers in comparison to single parallel executables. In particular, the invocation, pre- and post-processing, and monitoring of multiple concurrent jobs put a much higher load on the service nodes than the execution of single jobs. The service partition on ASCI Red was designed to manage basic coordination tasks and was not intended for significant floating point operations or heavy I/O demands. Although the individual processors are capable of computations, there were simply not enough service nodes to sustain significant activities. In the case of this design study, the service nodes were responsible for managing the optimization process (running the optimizer and querying for job completion) as well as managing concurrent simulation driver processes and parallel simulation monitoring processes. At certain points during the studies, Salinas jobs would hang on initiation. In the worst of these incidents, a service node became overloaded and crashed, which necessitated a full reboot of the machine.

The observed reliability problems stemmed more from the closely synchronized nature of concurrent simulation invocation than from the total amount of work being performed. In this study, it was found that staggering the Salinas job initiations by a few seconds allowed the load spreading utility sufficient time to spread the Salinas monitoring processes among the service nodes, which resulted in improved reliability. In addition, feedback to the ASCI Red system administration team since the conclusion of this study has resulted in several service partition reliability enhancements, including an increase in the number of service nodes from 16 (at the time of the study) to 52 (current number shown in Table 1).

6 Optimization Results

The objective of the optimization study was to satisfy safety margin requirements while remaining within a strict mass budget. These goals were achieved through an iterative, evolving process in which a total of four different optimization algorithms and two different optimization problem formulations were employed. Changes in method selection and problem formulation occurred as additional features of the problem became apparent. Comparisons between methods were not attempted as each study built on results obtained from previous studies. This approach is not uncommon in a results-driven application context for which problem characteristics are not known a priori. Having a toolbox of approaches available, as in DAKOTA, facilitates this type of evolving investigation.

6.1 Phase 1: Nongradient and Gradient-Based Optimization

The initial phase of the optimization study focused on the application of traditional nongradient and gradient-based optimization algorithms. These algorithms were provided in the SGOPT, NPSOL, and DOT optimization libraries available within the DAKOTA toolkit.

6.1.1 Coordinate Pattern Search Algorithm

The initial optimization formulation for the EP redesign was to maximize the minimum safety margin (SM), subject to constraints on the EP mass. A safety margin function was defined for each of the 55 blocks in the finite element model using the maximum response over all degrees of freedom in the block and over all 300 time steps in the transient simulation. Four shell thickness parameters and one foam density parameter from the EP model were selected as design variables for this optimization case. These five parameters were the most sensitive based on the sensitivity analysis study described in Section 2.

This optimization problem was formulated in DAKOTA as follows:

$$\begin{array}{ll} \text{maximize} & SM_{min} \\ \text{subject to} & 0.9M_{nom} \leq M \leq 1.1M_{nom}, \\ & \mathbf{x_L} \leq \mathbf{x} \leq \mathbf{x_U}, \end{array} \quad (1)$$

where SM_{min} is the minimum over all 55 safety margin values, M is the current mass of the EP, M_{nom} is the nominal mass of the EP, and $\mathbf{x}$ is the vector of 5 design variables with lower and upper bounds $\mathbf{x_L}$ and $\mathbf{x_U}$, respectively. The safety margin values were computed for the EP internal components based on either a stress allowable value or an acceleration allowable value. The safety margins based on stress values were computed as

$$SM_i = \frac{\sigma_i^a}{\sigma_i} - 1, \text{for } i = 1, \ldots, 42, \quad (2)$$

where σ_i^a is the allowable stress for the i^{th} block and σ_i is the computed maximum stress for all DOF in the i^{th} block for all time steps. Similarly, the safety margins based on acceleration values were computed as:

$$SM_i = \frac{g_i^a}{g_i} - 1, \text{for } i = 43, \ldots, 55, \quad (3)$$

where g_i^a is the allowable acceleration level for the i^{th} block and g_i is the computed maximum acceleration level for all DOF in the i^{th} block for all time steps. In both of these SM definitions (Equations 2 and 3), the fractional term is called the safety factor.

For this problem, σ_i^a was taken to be the yield stress for the particular material block and g_i^a was fixed at a constant value for all relevant material blocks. The nominal EP design had $SM_{min} = -0.48$, which indicates that some part of the EP was being exposed to twice the allowable stress/acceleration and was subject to failure.

Since this Phase 1 optimization formulation was expected to be nonsmooth due to switching among various components with the lowest safety margin, a nongradient-based method was selected for the initial optimization of the EP. This method was the coordinate pattern search method (CPS) contained in the Stochastic Global Optimization (SGOPT) software package (Hart (2001)). To incorporate the mass constraint, a simple penalty function was used, although this proved unimportant since the mass constraint never became active during the CPS iterations.

A single Salinas function evaluation required approximately 40 minutes on 256 processors of ASCI Red. Using the two-level parallel capabilities in DAKOTA, 10 instances of Salinas were executed concurrently. This completed a full optimization cycle of the CPS algorithm in one pass since CPS requires $2n$ function evaluations on each cycle (i.e., 10 Salinas jobs performed concurrently for $n = 5$

variables). The CPS method was able to improve the minimum safety margin from the nominal value of -0.48 to -0.21 with a mass increase of 5.4%, using a total of 171 function evaluations (Table 2). The pattern search made good progress until three separate margin functions were near the same minimum value (i.e., were active in defining SM_{min}) for the current design. This occurrence adversely affected the convergence rate of the pattern search method, as it was difficult to generate a step which simultaneously improved all three safety margins from the restricted set of coordinate search directions.

6.1.2 NPSOL SQP Algorithm

At this stage of the optimization, it was clear that obtaining a feasible design would be difficult with the CPS algorithm. Consequently, the problem formulation was changed to one that would be more amenable to gradient-based methods. In addition, more design freedom was added by introducing four new design parameters into the optimization problem. This new formulation of the optimization problem was

$$\begin{array}{ll} \text{minimize} & M \\ \text{subject to} & SM_i \geq SM_{target}, \text{for } i = 1, \ldots, 55, \\ & \mathbf{x_L} \leq \mathbf{x} \leq \mathbf{x_U}, \end{array} \quad (4)$$

where $SM_{target} = 0$, and $\mathbf{x}$ now contains 9 design variables. This formulation reduces nonsmoothness by eliminating the possibility of switching in the minimum safety margin function, as it allows the optimizer to track each of the 55 margin functions independently in the constraints. This does not totally eliminate all sources of nonsmoothness, however, since switching in space and time of the critical response within the context of a single margin function is still possible. Eliminating this final switching would have required separate constraints for each degree of freedom for each time step, or 150 million constraints. This was not practical for the optimization algorithms of interest.

The sequential quadratic programming (SQP) method in NPSOL (Gill *et al.* (1986)) was configured to use DAKOTA's parallel central finite differencing. For $n = 9$ variables, this gives a maximum concurrency of $2n + 1 = 19$ function evaluations. Given 10 concurrent Salinas executions on ASCI Red, the 19 jobs could be completed in two passes. Since NPSOL uses a gradient-based line search procedure (in user-supplied gradient mode), NPSOL avoids load imbalances in the line search phase. Starting from the best CPS design, NPSOL was able to improve the minimum safety margin from -0.21 to -0.15 and reduce the total mass to 4.4% over nominal, using a total of 114 function evaluations. Unfortunately, NPSOL was not able to run more than a few cycles before one of the Salinas jobs hung. This coincided with the expiration of the special allotment of 2560 ASCI Red processors that had been dedicated to this study.

6.1.3 DOT MMFD Algorithm

The optimization process was continued using 256 ASCI Red processors, i.e., a single Salinas function evaluation at a time. The decision was made to switch from NPSOL's SQP algorithm to DOT's Modified Method of Feasible Directions (MMFD) (Vanderplaats Research and Development (1995)) algorithm for two reasons. First, the DOT MMFD algorithm emphasizes finding a feasible point from the beginning of execution. In contrast, the NPSOL SQP algorithm is an infeasible method using an augmented Lagrangian merit function that will only satisfy the constraints at convergence. Second, the parallel load imbalance of DOT's value-based line search is not a hindrance when limited to a single Salinas function evaluation at a time.

Starting from the NPSOL best design point, the DOT MMFD algorithm improved the minimum safety margin value from -0.15 to -0.059, although it did not find a feasible design point. The mass increased to 5.3% over nominal. The DOT MMFD algorithm used 96 Salinas function evaluations before it was terminated.

6.1.4 Summary of Phase 1

Table 2 shows the progression of the optimization results for this study. The nongradient-based algorithm (SGOPT CPS) and the two gradient-based algorithms (NPSOL SQP and DOT MMFD) combined to move the infeasible nominal EP design to an improved infeasible design. The worst case safety margin violation had been reduced by approximately an order of magnitude, at a cost of a 5.3% increase in the mass of the EP. Figure 5 compares SM contours for the time step with the largest contrast between the nominal design and the best Phase 1 design.

At this point in the study, DAKOTA had controlled up to 10 concurrent Salinas jobs, each of

Table 2 The sequence of optimization results for the electronics package.

Design	Design Variables	Mass (kg)	SM Violations	Worst SM	Function Evaluations	Total Processors
Nominal		11.143	8	-0.480		
SGOPT CPS	5	11.747	4	-0.214	171	2560
NPSOL SQP	9	11.629	4	-0.147	114	2560
DOT MMFD	9	11.739	4	-0.0587	96	256
AO Verified	7	11.997	0	+0.0603	122	1024

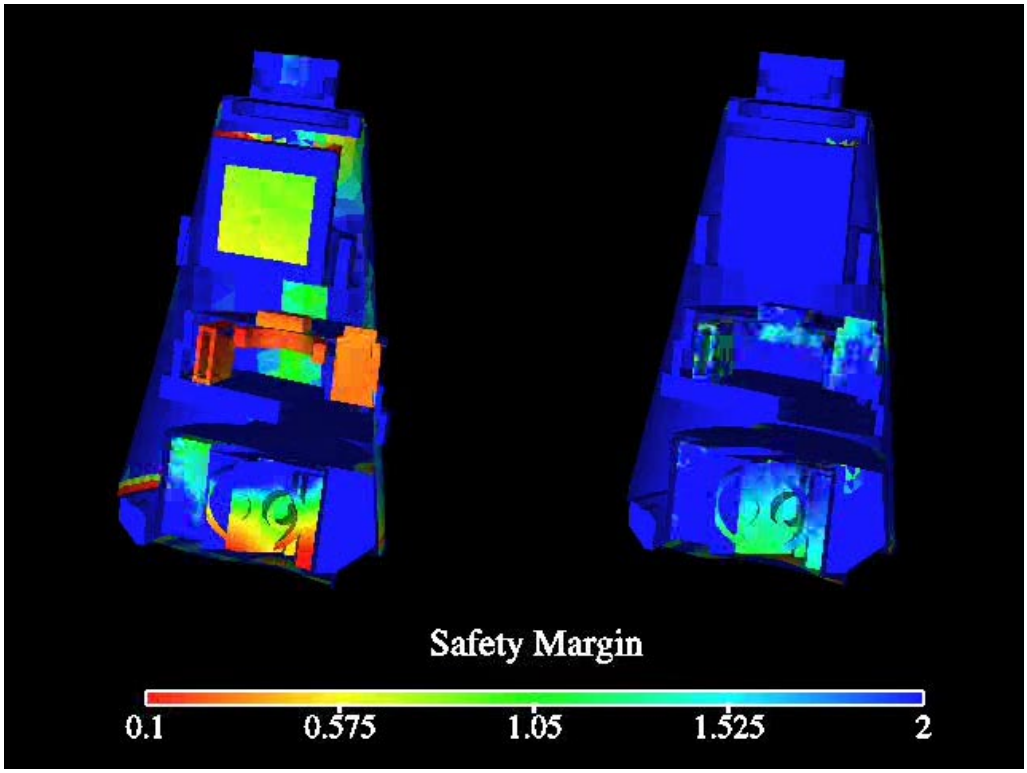

Fig. 5 A comparison of safety margin levels in the original electronics package model (left) and the optimized model from Phase 1 (right). The brighter colors indicate lower safety margins.

which used 256 processors. This use of up to 2560 processors was successful in compressing the duration of Phase 1 to four days. Without the use of parallel computing, equivalent calculations using serial optimization and serial simulation would have required in excess of 10 years to complete.

Additional information became available which motivated the next phase of this study. Some Salinas data from earlier parameter study runs were fully post-processed, and it was discovered that a subset of the safety margin functions exhibited considerable nonsmoothness (e.g., Figures 6, 7, and 8). One of these nonsmooth functions was active at the DOT MMFD solution and was inhibiting further progress. Thus, the decision was made to switch to an approximate optimization strategy that used surrogate models to smooth the noisy safety margin constraint functions.

6.2 Phase 2: Optimization using Approximate Models

The approximate optimization (AO) strategy used in this study is a simplified version of the surrogate-based optimization strategy described in Giunta and Eldred (2000). This AO strategy is divided into the following steps: (1) move limit (bounds) selection, (2) data sampling, (3) surface fitting to produce surrogate models, (4) optimization using the surrogate

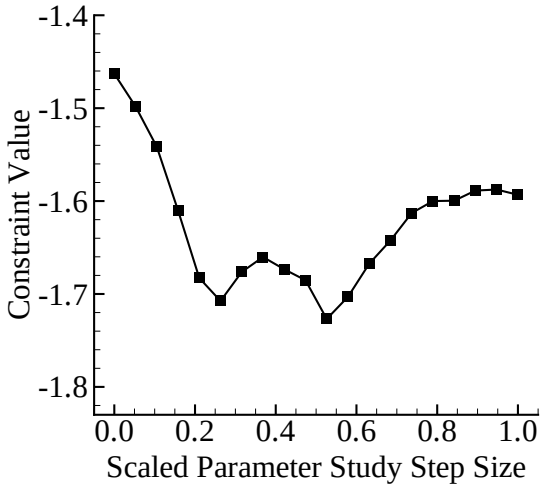

Fig. 6 Nonsmooth variations for constraint 29.

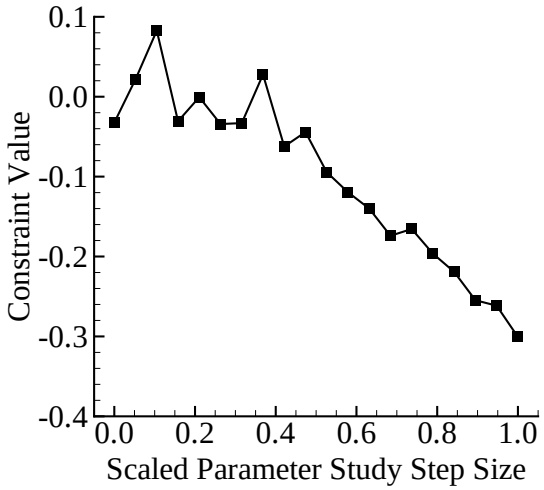

Fig. 7 Nonsmooth variations for constraint 52.

models, and (5) verification of the predicted optima. These steps are described below.

6.2.1 Move Limits

The best set of design variables found using DOT MMFD served as the starting design for the approximate optimization phase. An analysis of the previous optimization data showed that two of the variables did not strongly interact with the optimizer. Thus, these two variables were converted to constants, each having the optimal value obtained

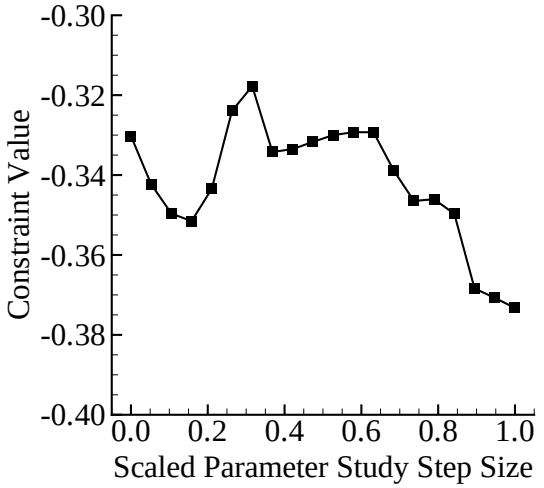

Fig. 8 Nonsmooth variations for constraint 55.

from the DOT MMFD results. The upper and lower bounds on each of the remaining seven variables were reduced to between 18% and 43% of the original bounds based on engineering judgment and the desire to balance the needs of sufficient design freedom and sufficient sampling density. In a formal trust region approach (Giunta and Eldred (2000)), the same upper and lower bound offsets would have been used for each variable; however, for a single AO cycle, custom bounds could be employed. For the remainder of this report, these bounds are referred to as the *move limits* of the approximate optimization.

6.2.2 Latin Hypercube Sampling

Next, the Latin hypercube sampling (LHS) method (McKay *et al.* (1979)) provided by the DDACE package (Martinez-Canales (2002)) within DAKOTA was used to generate 200 independent sample locations within the move limits. Salinas was used to evaluate as many of these EP designs as possible using the remainder of the computational budget devoted to this project. This Salinas/DAKOTA calculation again used two-level parallel computing, with four concurrent Salinas jobs each using 256 processors (1024 total processors).

Unfortunately, only 104 of the LHS design points were evaluated during the allocated ASCI Red computer time. While the 104 samples did not comprise a true LHS data set, this still provided sufficient sampling density to build the surrogate models. For example, 104 samples is sufficient to over-

fit a 7-dimensional quadratic polynomial (having 36 terms) by almost a factor of three. In addition, a statistical analysis was performed in order to check the distribution of the 104 samples in the design space. This analysis did not indicate any correlation or bias among the samples that would have rendered the Latin hypercube samples unusable.

6.2.3 Surrogate Model Construction

DAKOTA provides four global surrogate modeling techniques: (1) kriging spatial interpolation (Cressie (1991); Giunta and Watson (1998)); (2) quadratic polynomial regression (QuadPoly) (Myers and Montgomery (1995)); (3) multivariate adaptive regression splines (MARS) (Friedman (1990)); and (4) stochastic layered perceptron artificial neural networks (ANN) (Zimmerman (1996)).

The kriging, MARS, and ANN methods do not assume a particular trend in the data. That is, these three surrogate modeling methods can capture arbitrary variations in a given data set. In contrast, quadratic polynomial regression assumes that the data trends can be modeled using second-order functions. Thus, while all of these surrogate models provide a smooth functional form that is amenable to gradient-based optimization, the QuadPoly surrogate models enforce additional smoothing by nature of the assumed quadratic form.

6.2.4 Optimization with Surrogate Models

The results from the 104 Salinas jobs provided a set of mass and safety margin data which was used by DAKOTA to build 56 separate surrogate models. These surrogate models approximate the functional relationships between the objective and constraint functions (mass and 55 safety margins) and the seven EP design parameters. The surrogate models were used in the optimization problem in place of the Salinas simulations, thereby allowing multiple approximate optimizations to be performed at very low cost. The drawback is that the surrogate models can be inaccurate, particularly if the optimizer pushes the EP design near the move limit boundaries, where the surrogate models begin to extrapolate the data trends.

The first surrogate model type used in this study was quadratic polynomial regression. That is, the problem defined in Equation 4 was solved using QuadPoly surrogate models for mass and each of 55 constraints. For the initial approximate optimization case, the value of SM_{target} in Equation 4 was set to -0.05. Since these surrogate models allow for very inexpensive evaluations, Monte Carlo sampling studies were performed in order to identify good starting points (even though each function is unimodal, their intersections can produce multiple constrained minima), and then gradient-based optimizations were performed from these starting points. The bound constraints for both the Monte Carlo sampling and the gradient-based optimizations were identical to the move limit bounds used in the surrogate model construction. Next, SM_{target} was increased to 0.0 and the optimization was performed again. This sequence was continued with SM_{target} values of 0.05, 0.10, and 0.15. This was done to generate the mass versus safety margin tradeoff plot shown in Figure 9.

A similar sequence of approximate optimizations was performed for each of the other three surrogate model types: kriging, MARS, and ANN. In cases where the safety margin targets were not met, the target was reduced in an iterative fashion until a final maximized safety margin for the surrogate model was achieved. The EP mass versus safety margin tradeoff curves for these surrogate model types also are shown in Figure 9.

There are several interesting features to note about the trends in Figure 9. First, the ANN curve does not follow the same trends as the other three methods. This prompted an examination of the ANN algorithm in DAKOTA, and refinements to the ANN algorithm are planned. Second, the kriging and ANN tradeoff curves show kinks that result in an increased slope in mass versus SM. This behavior was traced to the optimizer bumping up against one or more of the move limit bounds, with a loss in design freedom resulting in a steeper mass versus SM trend.

6.2.5 Verification of Approximate Optima

The final step in the approximate optimization process was to run Salinas verification analyses for the EP designs identified in the approximate optimizations from the previous step. The best agreement between a predicted EP optimum and its Salinas verification analysis occurred for one of the designs predicted using the kriging surrogates. In this case, the actual mass was predicted very accurately (actual and predicted both 11.997 kg), and the actual

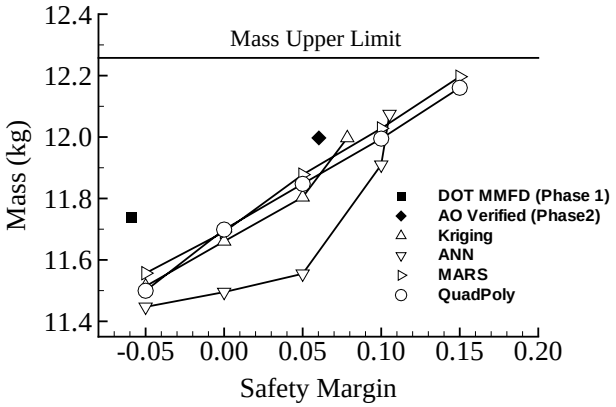

Fig. 9 Mass vs. safety margin tradeoff curves generated using various surrogate model types.

worst case safety margin value was +0.060 (predicted to be +0.078). The mass and worst case safety margin data for this AO verified design are listed in Table 2.

However, not all of the approximate optima were in such good agreement with the Salinas verifications. In some cases, the approximate optima had predicted a positive worst case safety margin, whereas the Salinas verification analysis yielded a negative worst case safety margin. This underscores the need for verification analyses whenever optimization is performed on surrogate models.

Had sufficient computational resources been available, this process would have been continued using a traditional trust-region surrogate-based optimization strategy (Giunta and Eldred (2000)) with additional rounds of sampling, fitting, optimizing, and verifying. This would mitigate the verification errors observed previously when only a single approximate optimization cycle is performed.

6.2.6 Summary of Phase 2

Phase 2 of the optimization study required 104 Latin hypercube samples and 18 verification analyses, for a total of 122 Salinas simulations. The use of up to 1024 processors to complete these analyses was successful in compressing the duration of Phase 2 to one additional day. Without the use of parallel computing, equivalent calculations on a single processor would have required an additional 2 years to complete.

7 Conclusions

This paper presents the results of a high-fidelity electronics package design study using a massively parallel structural dynamics code and a multilevel parallel optimization framework.

From the applications perspective, this study demonstrates the utility of having a toolbox of algorithms from which to tailor the optimization procedure as experience with a particular application increases. Through the combination of nongradient, gradient, and approximate optimization methods, the electronics package design was improved from an infeasible design which violated response allowables by a factor of two to a completely feasible design with positive design margins, while still remaining within strict mass targets. In retrospect, the approximate optimization techniques appeared to be the most effective in extracting the necessary trends from nonsmooth simulation results and would likely have reduced the overall computational expense if used from the beginning. In addition, these approximate techniques enabled the extraction of a design tradeoff curve of mass versus safety margin which proved useful in facilitating the design decision process.

From the parallel computing perspective, this paper validates the multilevel parallelism procedures in DAKOTA for a large-scale application and demonstrates the effectiveness of massively parallel computing in reducing the time to solve an actual engineering design problem. During the course of the EP study, a series of DAKOTA runs employed up to 2560 processors in a combination of coarse-grained and fine-grained parallel processing. These studies were completed in five days, where equivalent calculations on a single desktop computer would have required in excess of 12 years. Clearly, the effective use of massively parallel computing was a critical enabler in allowing a study of this magnitude.

While certain aspects of current-generation custom supercomputers do not yet lend themselves to routine studies of this type, several directions for improvement have been identified. In particular, exploiting tighter couplings between the optimization and simulation software will streamline process management and reduce the load on key supercomputer components. It is expected that advances in optimization and supporting parallel software will be successful in making high-fidelity studies of this type a standard component of modeling and simulation activities in the Department of Energy complex.

Acknowledgements The authors would like to express their thanks to Clay Fulcher for his assistance in formulating the design problem, to Ken Alvin for his assistance with modal sensitivity analysis and parameter screening, and to Garth Reese and Manoj Bhardwaj for their assistance with the Salinas simulation code. In addition, the authors appreciate helpful suggestions provided by Prof. Raphael Haftka of the University of Florida in reviewing this manuscript.

References

Biros, G., and Ghattas, O., 2000: Parallel Lagrange-Newton-Krylov-Schur Methods for PDE-Constrained Optimization. Part II: The Lagrange-Newton Solver, and its Application to Optimal Control of Steady Viscous Flows. Technical Report, Laboratory for Mechanics, Algorithms, and Computing, Carnegie Mellon University. (Available online from: http://www-2.cs.cmu.edu/~oghattas/)

Cressie, N., 1991: *Statistics for Spatial Data*, John Wiley and Sons, Inc., New York, pp. 1-26.

Day, D., 1998: A Basic Parallel Sparse Eigensolver for Structural Dynamics, Sandia Technical Report SAND98-0410C, Sandia National Laboratories, Albuquerque, NM.

Eldred, M.S., Giunta, A.A., van Bloemen Waanders, B.G., Wojtkiewicz, S.F., Jr., Hart, W.E., and Alleva, M.P., 2002a: DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis. Version 3.0 Users Manual. Sandia Technical Report SAND2001-3796, Sandia National Laboratories, Albuquerque, NM. (Available online from: http://endo.sandia.gov/DAKOTA/software.html)

Eldred, M.S., Giunta, A.A., van Bloemen Waanders, B.G., Wojtkiewicz, S.F., Jr., Hart, W.E., and Alleva, M.P., 2002b: DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis. Version 3.0 Reference Manual. Sandia Technical Report SAND2001-3515, Sandia National Laboratories, Albuquerque, NM. (Available online from: http://endo.sandia.gov/DAKOTA/software.html)

Eldred, M.S., Giunta, A.A., van Bloemen Waanders, B.G., Wojtkiewicz, S.F., Jr., Hart, W.E., and Alleva, M.P., 2002c: DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis. Version 3.0 Developers Manual. Sandia Technical Report SAND2001-3514, Sandia National Laboratories, Albuquerque, NM. (Available online from: http://endo.sandia.gov/DAKOTA/software.html)

Eldred, M.S., Hart, W.E., Schimel, B.D., and van Bloemen Waanders, B.G., 2000: Multilevel Parallelism for Optimization on MP Computers: Theory and Experiment, paper 2000-4818 in the *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA.

Farhat, C. and Roux, F., 1992: An unconventional domain decomposition method for an efficient parallel solution of large-scale finite element systems, *SIAM J. Sci. Statist. Comput.*, Vol. 13, No. 1, pp. 379-396.

Friedman, J. H., 1990: Multivariate Adaptive Regression Splines, *The Annals of Statistics*. (also published as Tech Report PUB-4960, Stanford Linear Accelerator Center, 1990)

Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., 1986: Users Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming, System Optimization Laboratory, TR SOL-86-2, Stanford University, Stanford, CA.

Giunta, A.A., and Eldred, M.S., 2000: Implementation of a Trust Region Model Management Strategy in the DAKOTA Optimization Toolkit, paper 2000-4935 in *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA.

Giunta, A. A., and Watson, L. T., 1998: A Comparison of Approximation Modeling Techniques: Polynomial Versus Interpolating Models, paper 98-4758 in *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, pp. 392-404.

Glass, G., 1993: *UNIX for Programmers and Users: A Complete Guide*, Prentice Hall, Englewood Cliffs, NJ.

Greenberg, D.S., Brightwell, R., Fisk, L.A., Maccabe, A.B., and Riesen, R.E., 1997: A System Software Architecture for High-End Computing, *Proceedings of Supercomputing 97*, San Jose, CA. (Available online from: http://www.supercomp.org/sc97/-proceedings/TECH/GREENBER/INDEX.HTM)

Hart, W. E., 2001: SGOPT User Manual, Version 2.0, Sandia Technical Report SAND2001-3789, Sandia National Laboratories, Albuquerque, NM. (Available online from: http://www.cs.sandia.gov/SGOPT/)

Heath, M.T., 1997: *Scientific Computing: An Introductory Survey*, McGraw-Hill, Boston.

Kernighan, B.W., and Ritchie, D.M., 1988: *The C Programming Language*, 2nd ed., Prentice Hall, Englewood Cliffs, NJ.

Martinez-Canales, M.L., 2002: DDACE – Distributed Design and Analysis of Computer Experiments (online document), Sandia National Laboratories, Livermore, CA. (Available online from: http://csmr.ca.sandia.gov/projects/ddace)

Maschhoff, K.J., and Sorensen, D.C., 1996: PARPACK: An Efficient Portable Large Scale Eigenvalue Package for Distributed Memory Parallel Architectures,

Applied Parallel Computing in Industrial Problems and Optimization, *Lecture Notes in Computer Science* (eds. J. Wasniewski, J. Dongarra, K. Madsen, and D. Olesen), Vol. 1184, Springer–Verlag, Berlin.

Mattson, T.G. and Henry, G., 1997: The ASCI Option Red Supercomputer, Intel Supercomputer Users Group, Thirteenth Annual Conference, Albuquerque, NM. (Available online from: http://www.cs.sandia.gov/ISUG97/papers/Mattson/-OVERVIEW.html)

McKay, M. D., Beckman, R. J., and Conover, W. J., 1979: A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code, *Technometrics*, Vol. 21, No. 2, pp. 239-245.

Myers R. H., and Montgomery, D. C., 1995: *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, Wiley, New York, pp. 79-123.

O'Neal, D., and Murgie, S., 2002: ANSYS Benchmarking Project: Evaluation of the Distributed Domain Solver (online document), National Center for Supercomputing Applications, Champaign, IL. (Available online from: http://www.psc.edu/~oneal/ansys/Article-DDS_Project_Benchmarks-ANSYS.pdf).

Reese, G.M., Bhardwaj, M.K., Driessen, B., Alvin, K.F., and Day, D., 2000: Salinas - An Implicit Finite Element Structural Dynamics Code Developed for Massively Parallel Platforms, paper 2000-1651 in *Proceedings of the 41st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Atlanta, GA.

Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J., 1996: *MPI: The Complete Reference*, MIT Press, Cambridge, MA.

Sobieszczanski-Sobieski, J., Kodiyalam, S., and R.-J. Yang, 2001: Optimization of Car Body Under Constraints of Noise, Vibration, and Harshness (NVH), and Crash, *Structural and Multidisciplinary Optimization*, Springer, Volume 22, Issue 4, pp. 295-306.

Stroustrup, B., 1991: *The C++ Programming Language*, 2nd ed., Addison-Wesley, New York.

Thimbleby, H., 1993: Computerized Parkinson's Law, *Computing and Control Engineering Journal*, Vol. 4(5), pp. 197-198.

Tomkins, J.L., 1996: The ASCI Red TOPS Supercomputer (online document), Sandia National Laboratories, Albuquerque, NM. (Available online from: http://www.sandia.gov/ASCI/Red/RedFacts.htm)

Tuminaro, R.S., Heroux, M.A., Hutchinson, S.A., and Shadid, J.N., 1999: Official Aztec User's Guide: Version 2.1, Sandia Technical Report SAND99-8801J, Sandia National Laboratories, Albuquerque, NM. (Available online from: http://www.cs.sandia.gov/CRF/Aztec_pubs.html)

Vanderplaats Research and Development, Inc., 1995: *DOT Users Manual, Version 4.20*, Colorado Springs, CO.

Venkataraman, S., and Haftka, R.T., 2002: Structural Optimization: What Has Moore's Law Done For Us?, paper 2002-1342 in *Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Denver, CO.

Yang, R.-J., Akkerman, A., Anderson, D. F., Faruque, O. M., and Gu, L., 2000: Robustness Optimization for Vehicle Crash Simulations, *Computing in Science and Engineering*, IEEE Press, Vol. 6, Issue 2, pp. 8-13.

Zimmerman, D., 1996: Genetic Algorithms for Navigating Expensive and Complex Design Spaces, Final Research Report prepared for Sandia National Laboratories (technical contact M. Eldred).